Metrics for BPEL Process Context-Independency Analysis

A. Khoshkbarforoushha[1], P. Jamshidi[2], A. Nikravesh[1], F. Shams[1]

[1]*Automated Software Engineering Research Group,*

Electrical and Computer Engineering Faculty,

Shahid Beheshti University GC, Tehran, Iran.

{a_khoshkbarforoushha, a_nikravesh, f_shams}@sbu.ac.ir

[2]*Lero - The Irish Software Engineering Research Centre,*

School of Computing, Dublin City University, Dublin, Ireland

pooyan.jamshidi@computing.dcu.ie

The preliminary version of this paper presented at IEEE International Conference on Service-Oriented Computing and Applications, 2009 (SOCA'09), Taipei, Taiwan. DOI: 10.1109/SOCA.2009.5410260

Abstract

BPEL processes are workflow-oriented composite services for service-oriented solutions. Rapidly changing environment and turbulent market conditions require flexible BPEL processes to adapt with several modifications during their lifecycles. Such adaptability and flexibility requires the low degree of dependency or coupling between a BPEL process and its surrounding environment. In fact, heavy coupling and context-dependency with partners provoke several undesirable drawbacks such as poor understandability, inflexibility, inadaptability, and defects. This paper is to propose metrics at the design phase to measure BPEL process context-independency. With the aid of these metrics the architect could analyse and control the context-independency of a BPEL process quantitatively. To validate the metrics, authors collected a data set consisting 70 BPEL processes and also gathered the expert's rating of context-independency through conducting a controlled experiment. The obtained results reveal that there exists a high statistical correlation between the proposed metrics and the expert's judgment of context-independency.

Keywords *BPEL Process Coupling Measurement. Service Coupling Metrics. Composite Service Context-independency. Service-oriented Metrics. SOA Coupling Metric. Workflow Metrics.*

Introduction

The unpredictability of business processes and turbulent business environment causes enterprise's processes to change dramatically. In order to meet these modifications, enterprises require maintainable workflow systems with the ability to dynamically adapt to the changing environment [1][2]. This adaptability actually refers to the flexibility of the business processes. The flexibility of a business process is characterized by capability to readily adapt to new, different, or changing requirements [3]. Such adaptability requires the low degree of dependency or coupling between a business process and its changing environment. In this regard, solutions that do not exhibit low coupling might suffer from the following developmental difficulties [4]. Firstly, any changes in one module provoke ripple effects in the other modules. Secondly, modules are difficult to understand in isolation. Finally, their testability and reusability are reduced because dependent modules must be included. These independent modules in service-oriented architecture (SOA) terminology are referred as loosely-coupled services. Loosely-coupled services are those that expose only the necessary dependencies. This is particularly important when services are subject to changes repeatedly. Minimal dependencies assure that there will be a minimal amount of changes to other services when one service is modified. Such an approach not only improves robustness, but makes systems more resilient to changes [5].

Many researchers have attempted to develop a single metric for comprehensive measurement of software coupling [4]. In this regard, there is a large body of research and practice on metrics for traditional software development paradigm such as Object-Oriented (OO) or Component-Based Development (CBD); however they are not applicable to service-oriented solutions. This is mostly due to three reasons [6]: a) Service-oriented computing is founded on independent, platform agnostic services that coupled with other services through only their interfaces; b) Services can be realized by elements belonging to various development paradigms or languages; c) Services are in higher level of abstraction comparing to components and objects. In contrast to OO and CBD paradigms, the work on service-oriented paradigm metrics particularly business process coupling and context-independency are very limited. On the other hand, heavy coupling and context-dependency of business processes with its surrounding environment provokes several undesirable drawbacks such as poor understandability, inflexibility, and defects. Therefore, heavy dependency and coupling should be avoided. Furthermore, it is important to develop metrics to help architect analyze the context-independency of business processes quantitatively. After that they can be reengineered to reduce the context-dependency and increase flexibility.

Many efforts have been made by the service-oriented community to define standards and protocols in order to enable flexible business process management solutions. In this regard, BPEL [5] as a de facto standard provides a workflow-oriented composition model for service-oriented solutions that facilitates the system integration through orchestration and choreography of services. Business processes which are particularly under focus of this paper for context-independency exploration are, in fact, BPEL processes. In this regard, the major objective of this paper is to introduce a novel quantitative metric to measure BPEL process context-independency. By adopting such a metric, SOA architect will be able to determine to what extent a BPEL process is context-independent. This serves as a foundation for quality service-oriented solution design.

The approach of the paper for predicting context-independency is based on measuring the coupling value between a BPEL process and its partners (i.e. web services). The coupling value of

a certain BPEL process is examined and quantified on the basis of measuring its interaction activities (e.g. <invoke>, <reply>, <receive>) within structured activities (e.g. <sequence>, <pick>). In the proposed metrics in addition to the number of invocations, exchanged messages between a BPEL process and its partners, synchronicity or asynchronicity, and the role of partner links are considered as context-independency factors.

A data set consisting 70 professional services which exhibited different levels of coupling as reflected by the metrics was collected. Then a workshop comprises 20 participants who were asked to rate the potential coupling of the services was then held to provide empirical data. The correlation between the independent and dependent variables was then assessed using statistical methods. The majority of the obtained results support our contribution, indicating a statistically significant relationship between context-independency of the services and expert's rating.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 explores BPEL process context-independency in terms of its definition and motivation. Section 4 presents context-independency metric for BPEL processes. Section 5 explores interaction patterns against proposed metrics. Section 6 evaluates and explains context-independency metric using both an exemplary scenario and implementing a recursive algorithm for automating metric measurement. Section 7 presents both theoretical and empirical validation of the metrics and their concerned results. We next sum up the discussion and provide some conclusions in Section 8 and 9, respectively.

Related work

In software engineering, many quality metrics have been proposed in order to have better structures in design. This naturally leads to a more understandability of the logic in addition to easier module identification and a higher degree of maintainability and configurability [7]. The design quality is related to five design principles: cohesion, coupling, complexity, modularity, and size [7]. Coupling is a more important factor than cohesion when estimating some of the external quality characteristics of software elements [8]. Coupling concept in the context of procedural systems is defined as the measure of the strength of association established by a connection from one element to another. In the context of this paper, coupling is defined as the strength of associations established by a connection from a BPEL process to partner links. Since context-independency is inherently the inverse of coupling (context-dependency), the focus of this section is mainly on coupling as an important measure for quality software engineering, irrespective of the design approach.

Stevens et al. [9] have introduced different types of coupling in procedural systems such as content, data, control, and message coupling. Due to the existence of different mechanisms that can constitute coupling in OO systems (direct coupling, coupling via inheritance, polymorphic coupling, etc.), coupling has more dimensions and is more complex to be measured comparing to the procedural ones [8]. Currently, there are three major frameworks characterizing various dimensions and types of OO coupling [8]. However, coupling metrics in OO and procedural software development methods are not applicable in service-oriented systems. This is mainly due to the differences in the structure of service-oriented and OO/Procedural systems [6]. Although design principles of SOA have been fully discussed in several research reports, most of them do not explain how these principles can be quantitatively measured and many of the proposed measures have not been fully validated in real SOA designs.

In order to calculate coupling, Perepletchikov et al. [8] have focused on measuring the number and type of the connections between various services and defined nine types of coupling metrics. These metrics mostly are related to service implementation elements, assuming different weight factors for the relationships between elements. An aggregation of these metrics was used to define coupling at the service level. Unlike [8], [10] defines coupling measures assuming the availability of only service design level information. Sindhgatta et al. [10] believe that in a service-oriented design, there is a need to distinguish between two categories of coupling which are the dependence of a service on the other services, and its dependence on messages. Accordingly, they define Service Operational Coupling Index (SOCI) and Inter-Service Coupling Index (ISCI) metrics for first category and Service Message Coupling Index (SMCI) metric for message coupling category. This work also explores service coupling generally. Indeed, they do not differentiate between different kinds of services including atomic, composite, or workflow-oriented business services in SOA layers. Besides, their work neglects the role of enabling technologies (e.g. Web services, BPEL, etc.) in coupling value calculation.

Pautasso et al. [11] have mentioned coupling as a complex concept that requires exploring a multidimensional space. Hence, they have collected 12 facets for coupling with respect to specific

aspect of web technologies. In addition, they have discussed relationships and interdependencies between those facets and proposed a multi-facet definition to evaluate the coupling of the existing web technologies and web service frameworks (i.e. RESTful HTTP, RPC over HTTP, WS-*/ESB) but they have not proposed any quantitative formulas to measure coupling value of a given service-oriented design.

Xu et al. [12] have proposed a decoupling metrics based on seven parameters, namely: service stateness (stateless/stateful), interactions, interface, invocation models, self-containment, implicit invocation, and biding models. Although their metrics are applicable to measure coupling of a service-oriented design, they neglect business processes in their calculations. To be more specific, they focus on services generally while in specific process-centered services some of the above-mentioned parameters could not be utilized. For example, unlike web services, BPEL processes are stateful long-running interactions, hence such a parameter is not applicable to workflow-oriented web services.

Reijers et al. [13] developed an extension of the formalization of the information element structure and the cohesion metrics, as presented in earlier work [14], and defined new metrics for coupling in workflow processes. Process coupling is defined formally as a function of the number of connections between two distinct activities and the number of activities within the process. Since the coupling metric in [13] did not consider resource issues, the authors extend their work in [7] to alleviate the issues. Vanderfeesten et al. [15] extends other work by specifically incorporating the effects of different types of connectors used on a process model's coupling level. As an extension to [7], Vanderfeesten et al. [15] propose a heuristic that offers guidance for the creation and evaluation of process designs, in which, complication of connections between the activities is considered and a tool support for these metrics is provided as well. These heuristics can be used to select from several alternatives of the process design that is strongly cohesive and weakly coupled.

To make conclusions on the reviewed related work, we have to emphasize that most of the SOA-related metrics lack enough attention to the BPEL processes that consume services and at the same time act as an underlying basis for service orchestration and composition. Although, as cited before, there are some work that investigate coupling concept in workflows, but they are different from our work from two perspectives. Firstly, most of the work especially [7][13][15] focus on how strongly the activities in a workflow process are related, or connected, to each other, while in our work we investigate the measure of the strength of association established by a connection from a process model to its surrounding environment. Secondly, none of the works in the field of coupling metrics for SOA systems considered BPEL structure or any other design/implementation alternative structure. On the other hand, process/workflow coupling metrics do not consider SOA-related technologies such as BPEL composition structure or the underlying service-oriented implementation technologies and standards. Another important issue is that some of the metrics provided in some research works are not concrete enough, and their validation methods are relatively immature, mostly based on providing examples or carrying out simple case study, analogies and comparisons.

BPEL Process Context-Independency

BPEL Process

In BPEL, a business process is a coarse-grained web service (i.e. composite web service) executing a control flow to complete a business goal. Each BPEL process consists of various steps. Each step is called an activity. BPEL activities divided into Basic and Structured categories [5]. Basic activities are used for common tasks e.g. invoking a web service or manipulating data. The important basic activities are `<invoke>`, `<receive>`, `<reply>`, and `<assign>`.

Structured activities are used for arranging the structure of BPEL process. Structured activities can contain both basic and structured activities in order to implement complex business processes. The structured activities are `<sequence>`, `<flow>`, `<switch>`, `<while>`, and `<pick>`. `<onMessage>`, `<onAlarm>` activities are used within the `<pick>` construct to capture events either message-based or time-based.

In addition to basic and structured activities, we require a preliminary knowledge on how BPEL links to its partners, manages events, or handles faults in order to investigate context-independency concept in any circumstances.

BPEL calls the links to all parties it interacts with partner links. Partner links can be links to web services that are invoked by the BPEL process or links to clients, and can invoke the BPEL

process. Partner links as concrete references to parties that a BPEL process interacts with are defined using <partnerLink> construct which is nested within the <partnerLinks>. Every <partnerLink> has an attribute that define the *role* of BPEL process with its partner and vice versa.

Even though the <pick> activity can be used for managing events, but sometimes we would like to react on event while the BPEL process executes. In this regard, <eventHandlers> and <onMessage> constructs are utilized.

Fault handling is an important aspect of BPEL process. In BPEL faults can be sourced from WSDL faults, error conditions in run-time environment, and so on. To react to faults that occur while the BPEL process activities are executing, <faultHandlers>, <catch>, <catchAll> constructs are used.

Definition of Context-Independency

Context-independency is defined as the extent to which a BPEL process requires the knowledge of its surrounding environment [17]. It is also called loosely-coupled [18]. An environment, which a business process needs to interact with and acquire or deliver the required information, consists of involved partners such as web services and clients.

Is Context-Independency a necessity?

Regarding to context-independency definition, a context-independent system is expected to have the least coupling with its surrounding environment, in other words, context-independency is directly related to, and is synonymous with, loose-coupling.

Loose-coupling directly affects maintainability. It is an attribute of systems, referring to an approach to designing interfaces across modules to reduce the interdependencies across modules, and reducing the risk that changes within one module will create unanticipated changes within other modules [4]. In other words, loosely-coupled systems are least exposed to ripple effects caused by a change made or a fault/failure happened somewhere in the system. Therefore, this feature helps to have higher availability.

Loose-coupling provides extensibility to the design [19]. It specifically seeks to increase flexibility in adding, replacing, and changing modules. It also helps to reduce the overall complexity and dependencies, makes application landscape more agile and enables quicker change.

Loose-coupling also affects security. In this regard, authors of [20], state that with a loose-coupling approach, security can be designed as a set of services that are independent of any other application. Indeed, security policy can be implemented once as a service and linked to each application that it applies to. This means that businesses can have better ways to ensure control over security. Moreover, when implicit dependencies between services that exist because of the coupling are reduced the testability and reusability are increased [4].

Loosely-coupled systems have the added advantage that they tend to be built more quickly. This is due to the low amounts of inter-module dependency. However, when making architectural decisions, one must carefully analyze the advantages and disadvantages of the level of coupling [21]. Loose-coupling is not always positive. For example, if systems de-coupled when using message-oriented middleware, it is tough to provide transactional integrity. Indeed, data replication across different systems provides loose-coupling, but provokes issues in maintaining synchronization [19].

BPEL Process Context-Independency Metric

Measurement strategy

There are two kinds of measurement strategies including assessment and prediction. Measurement for assessment is helpful in understanding what exists now or what has happened in the past, while prediction deals with measuring attributes that does not exist yet [22]. Our metrics are expected to be leveraged at analysis and design phase of SOA development. At this phase any implementation does not exist such that we are just able to predict the context-independency attribute. Therefore, the measurement strategy for introducing context-independency metric is prediction.

Another point about the measurement approach that should be elucidated is if it is direct or indirect. Direct measurement of an element's attribute involves no other elements or attributes. On the contrary, indirect measurement is dependent on some other attribute measures [22]. With respect to the point that our metrics are to predict context-independency of a given BPEL process based on coupling measures, they fall into the indirect measurement category.

The third point of our measurement approach is scale. Measurement scale refers to measurement mapping together with empirical and numerical relation systems. There are five measurement scales including nominal, ordinal, interval, ratio, and absolute. Each of these scales has specific characteristics. Since our metrics preserve ordering, the size of intervals between elements, ratios between elements, and start from zero they can be considered as ratio type.

Definition of Context-Independency Metric

In our perspective, context-independency can be estimated through coupling concept, as Vanderfeesten et al. [15]emphasize that the coupling measures the number of interconnections between the activities in a process model. On the other hand, a BPEL process is dependent to a context to the extent that is coupled with its web services, resources, and clients. Hence, service coupling measurement paves the way for context-independency estimation. Moreover, service coupling measurement provides a nontrivial insight into the communication overhead of discovered composite services with its context. In this regard, Papazoglou and van den Heuvel[23] state that the number of messages exchanged between a sender and addressee should be minimal.

A composite service, which is realized through BPEL, communicate and interact with involved partners (i.e. web services, clients, etc.) by means of `<invoke>`, `<reply>`, `<receive>`, and `<onMessage>` activities. We refer to these activities as interaction activities in the rest of the paper.

The source of coupling between a BPEL process and its surrounding environment is interaction activities. Through these activities the process sends to or receives messages from synchronous or asynchronous operations of the partners. These messages are those the operations receive as inputs, interpret and process, and those they need to produce as output, as declared in the interface. This means, the interaction activities impose various degree of coupling weights since the exchanged messages have different complexity. In fact, the more complex message requires the more knowledge to handle that. In BPEL, messages are stored in variables. Each variable could store WSDL messages, XML Schema type either simple or complex, and XML Schema element. To uniform the calculation, we consider them as a different collection of data fields. For instance, a PatientRequest messages contains some data fields such as PatientID, Firstname, Lastname, and so on. In our perspective, the basic coupling value for each interaction activity is the number of data fields, since the more data fields with diverse data types impose more coupling with partners. Thus, the coupling value of interaction activities is calculated as follows:

Let.

- C_{ia}: refers to the coupling value of interaction activities (ia).
- IVDF: refers to the number of Input Variable Data Fields.
- OVDF: refers to the number of Output Variable Data Fields. This parameter exists in case of synchronous invocation.
- VDF: refers to the number of Variable Data Fields.

$$C_{ia_j} = \begin{cases} (IVDF_j + OVDF_j) & ia_j = \langle invoke \rangle \\ VDF_j & ia_j = \langle receive \rangle \; or \; \langle reply \rangle \; or \; \langle onMessage \rangle \end{cases} \quad (1)$$

In equation 1, the interaction activities are divided into two categories, since the *invoke* construct almost has both input and output variables. The reason is the *invoke* construct provides a kind of two-way connection. In contrast to *invoke* construct, *receive*, *reply*, and *onMessage* provide one-way connection.

The coupling of a BPEL process with its partners becomes significant on the condition that the interaction activities incorporated within the structured activities. For instance, `<flow>` construct may contain four *invoke* activities which should be executed in parallel. Therefore, the coupling of interaction activities must be computed as they are incorporated into structured ones in the following subsections.

Sequence activity

A <sequence> activity is used to define activities that need to be performed in a sequential order. In BPEL, <sequence> construct is represented as follows:

```
<sequence standard-attributes>
  standard-elements
  activity+
</sequence>
```

The coupling of BPEL process with <sequence> activity is calculated as follows:

$$C_{BPEL}(sequence) = \sum_{j=1}^{n} C_{ia_j} \qquad (2)$$

Where

- C_{BPEL}(sequence): refers to the coupling value of a BPEL process with a *sequence* construct.
- *n* is the number of interaction activities within a sequence.

In fact, the sequence construct does not impose any further coupling to the interaction activities.

Switch activity

To express conditional behavior, the <switch> activity is used. It consists of one or more conditional branches defined by <case> elements, followed by an optional <otherwise> element. The case branches of the *switch* are considered in alphabetical order. The switch activity has the following structure:

```
<switch standard-attributes>
  standard-elements
  <case condition="bool-expr">+
    activity
  </case>
  <otherwise>?
    activity
  </otherwise>
</switch>
```

The coupling of BPEL process with <switch> activity is computed as follows:

$$C_{BPEL}(switch) = \sum_{i=1}^{m} \left(\frac{1}{m} \times \sum_{j=1}^{n} C_{ia_j} \right)_i \qquad (3)$$

Where

- *m* is the number of switch conditions.
- *n* is the number of interaction activity within condition *i*.

In our view, coupling of interaction within a *switch* construct is related to the probability of the occurrence of each of its conditions. Thus, the coupling of switch construct is calculated as the summation of the probability of each condition occurrence multiply with the number of interaction activities within that condition. In run-time, the probability of execution for every single conditional branch may differ from the other branches. Such information can change the value of context-independency. However, due to the fact that the metric is in design-time we assume that the probability of execution for branches is equivalent.

Pick activity

The <pick> activity is used to wait for the occurrence of one of a set of events and then perform an activity associated with the event. The structure of this construct is as follows:

```
<pick createInstance="yes|no"? standard-
attributes>
  standard-elements
  <onMessage partnerLink="ncname"   portType="qname"
          operation="ncname" variable="ncname"?>+
    activity
  </onMessage>
  <onAlarm (for="duration-expr" | until="deadline-expr")>*
    activity
  </onAlarm>
</pick>
```

The coupling of BPEL process with <pick> activity is computed as follows:

$$C_{BPEL}(pick) = \begin{cases} \sum_{i=1}^{m}\left(\frac{1}{m}\times\sum_{j=1}^{n} C_{ia_j}\right)_i & if\ \langle onAlarm\rangle\ triggers. \\ \sum_{i=1}^{m}\left(\frac{1}{m}\times\sum_{j=1}^{n} C_{ia_j}\right)_i + C_{ia_l} & if\ \langle onMessage\rangle\ triggers\ and\ ia_l = \langle onMessage\rangle \end{cases} \quad (4)$$

Where

- *m* is the number of <onAlarm> and <onMessage> constructs.
- *n* is the number of interaction activities within <onAlarm> or <onMessage> constructs.

The semantic and behaviour of *pick* construct is similar to *switch* activity. In equation 4, if an <onMessage> receives the expected event, we add the coupling value of that construct to the total coupling value. This is why unlike *onAlarm* activity, the *onMessage* is a kind of interaction activity and itself imposes some dependency to the context too.

Flow activity

The <flow> activity provides concurrent execution of enclosed activities. The *flow* activity also allows the synchronization of activities within the *flow*. To do so, <Link> construct specifies a dependency between a source and target activity. The syntax of flow activity is as follows:

```
<flow standard-attributes>
  standard-elements
  <links>?
    <link name="ncname">+
  </links>
  activity+
</flow>
```

The coupling of BPEL process with <flow> activity is computed as follows:
Let.

- *n:* refers to the number of interaction activities within a flow construct.
- *NL*: refers to the Number of *Link* construct in a BPEL process.

$$C_{BPEL}(flow) = \begin{cases} \left(\sum_{j=1}^{n} C_{ia_j}\right)^2 + \sum_{j=1}^{n} C_{ia_j} \ \Big/ 2 & the\ flow\ construct\ with\ concurrency\ behavior. \\ \sum_{j=1}^{n} C_{ia_j} + \ NL & the\ flow\ construct\ with\ synchronizing\ behavior. \end{cases} \quad (5)$$

Flow construct provides a kind of parallel interactions with the partners. Although the activities within a *flow* construct are triggered at the same time but they are terminated at different moments.

Consequently, comparing to <sequence> construct, the *flow* imposes significant coupling with partners. In other words, the number of interactions with the partners is not constant and the coupling value varies during the time. Suppose that there are three interaction activities within a *flow* that are expected to be executed concurrently. C_{ia} value for three activities are the value of 1. They are triggered at time t_0. Suppose that in t_1 one of the activities terminated. Similarly, in t_2 and t_3 the two other activities terminated respectively. Thus, before t_1 the coupling value is 3, and before t_2 and t_3 the coupling value is 2 and 1 respectively. As a result, the total coupling value is 6. Based on what was elaborated, first part of equation 5 demonstrates the formula for the *flow* construct with concurrency behaviour.

The second part of equation 5 refers to a *flow* activity with synchronizing behaviour. In this case, the interaction activities are executed in terms of defined dependency flow between them. Thus, the coupling value is the summation of coupling value of interaction activities in addition to the number of *link* between *source* and *target* activities. The reason is the <Link> construct impose a kind of coupling between the activities that must be considered in total coupling value.

While activity

A <while> activity is used to define an iterative activity. The iterative activity is performed until the specified Boolean condition no longer holds true. In BPEL, the representation of *while* construct is as follows:

```
<while condition="bool-expr" standard-attributes>
  standard-elements
  activity
</while>
```

The coupling of BPEL process with <while> activity is computed as follows:

$$C_{BPEL}(while) = N \times \sum_{j=1}^{n} C_{ia_j} \qquad (6)$$

Where

- *n* is the number of interaction activities within *while* construct.
- *N* refers to the number of loop iterations.

Process coupling with its partners is calculated as the number of <while> iterations multiply with the number of interaction activities within that. Since our metrics are at the design time, we are not able to calculate the iteration numbers exactly. However, we could estimate the number of loop iteration with the aid of environmental data about the business process that can be collected from domain experts.

Context-Independency Measurement

Taking all the metrics of the constructs into account, coupling and context-independency of the BPEL processes could be measured as follows:

Let:

- $C_{BPEL}(BPEL)$: refers to coupling measurement of a BPEL process.
- SA_i: refers to ith Structured Activity within a BPEL process.
- *j*: refers to the number of structured activities.
- *p*: is constant to 1.
- CI_{BPEL}: refers to context-independency estimation of a BPEL process.
- R_{PL}: refers to the number of Partner Link Roles.
- *n*: refers to the number of partner links.

$$C_{BPEL}(BPEL) = \left(\sum_{i=1}^{j}\left(C_{BPEL}(SA_i)\right)\right) \times \left(\sum_{k=1}^{n}\left(\frac{1}{R_{PL_k}}\right)\right) \qquad (7)$$

$$CI_{BPEL} = \frac{p}{C_{BPEL}(BPEL)} \quad (8)$$

Equation 7 is the summation of service coupling with its partners which is provoked by interaction activities multiply by summation of partner link role number. However, this question may arise "why the number of partner links should be taken into account". In our perspective, the context-independency of a BPEL process should be normalized on the basis of partner links. To elaborate more, suppose that there are two BPEL processes. The first one, invoke 3 operations of a web service while it is executing. On the contrary, the second BPEL process invokes 3 operations from three different web services. Apparently, in the latter case the process require more knowledge from its surrounding environment comparing to the former. Consequently, we multiply the coupling value of interaction activities within structured constructs by the number of partner link roles.

The second point is during the BPEL process execution different operations either synchronous or asynchronous are invoked. If we invoke a synchronous operation, the BPEL process waits for the reply while in asynchronous operation invocation process could do some other processing steps between the `<invoke>` and `<receive>` rather than waiting for the reply. Therefore, asynchronous invocation has lower coupling between service requester and service provider than synchronous invocation. In BPEL a partner link type must have at least one role and can have at most two roles. In asynchronous relation there are two roles: one defines the role of web service to the BPEL process and the other defines the role of BPEL process to the web service. Thus the second part of equation 7 takes the value of 1 for synchronous relation and the value of 0.5 for the asynchronous relation. This means, the equation imposes higher coupling value for synchronous operations comparing to the asynchronous one.

Finally, based on what was discussed in previous sections, the relationship between service coupling and service context-independency is reverse, such that service context-independency is quantified via equation 8.

Interaction patterns

Every possible metric for BPEL process coupling and subsequently context-independency measurement must be evaluated against various kinds of interaction patterns. As specified in [24], there are ten interaction patterns as follows:

- One-Way Message
- Synchronous Interaction
- Asynchronous Interaction
- Asynchronous Interaction with Timeout
- Asynchronous Interaction with a Notification Timer
- One Request, Multiple Responses
- One Request, One of Two Possible Responses

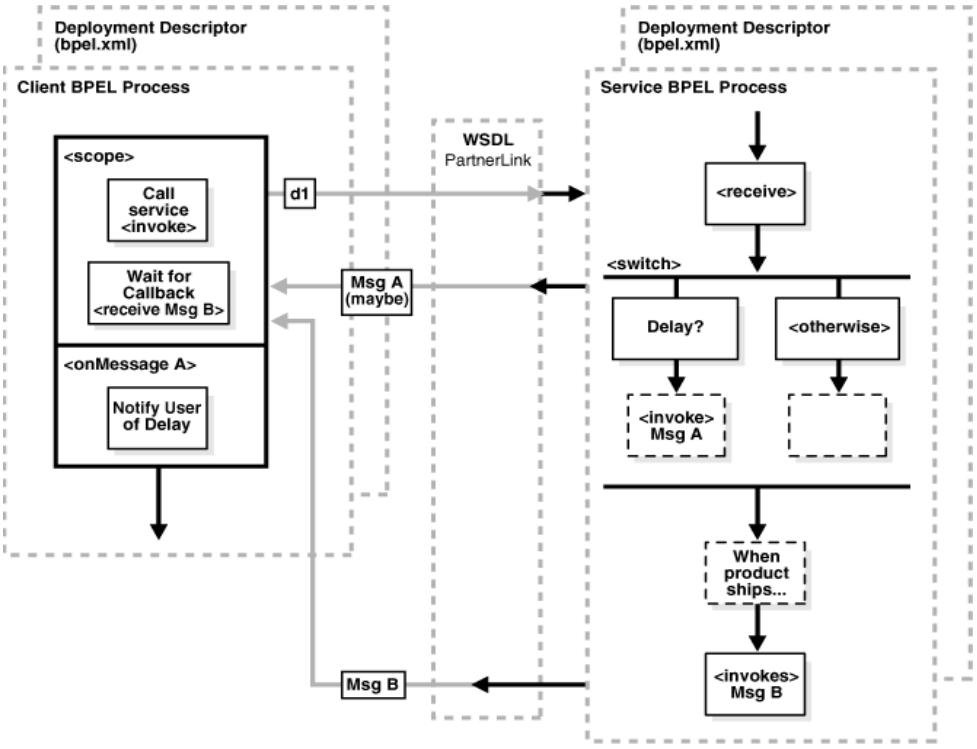

Figure 1: One Request, a Mandatory Response, and an Optional Response [24].

- One Request, a Mandatory Response, and an Optional Response
- Partial Processing
- Multiple Application Interactions

In the same way, [25] enumerated 13 patterns in four categories including Single-transmission bilateral interaction patterns, Single-transmission multilateral interaction patterns, Multi-transmission interaction patterns, and Routing patterns.

The proposed metric for context-independency prediction remarkably supports these patterns, even though they must be evaluated separately with different context-independency values. This is due to the fact that our metric is on the basis of interaction activities.

To illustrate the point, we apply the metric to one of the complicated patterns that is *one Request, a Mandatory Response, and an Optional Response* pattern, as shown in Figure 1.

In this interaction, the client sends a single request to a service and receives one or two responses. Suppose that the request is to order a product online. If the product is delayed, the service sends a notification message to the customer. In any case, the service sends a notification when the item ships.

In the service BPEL process side there are two structured activities. The first one is `<sequence>` (a BPEL process will have the top-level element which is usually `<sequence>`). The second one is `<switch>` construct which is inside of the `<sequence>` activity. Within these structured activities there are three interaction activities including one `<receive>` and two `<invoke>` constructs. After such analysis, we are able to estimate process context-independency via the proposed formulations.

Since the WSDL code is not available, it is assumed that the C_{ia} for the above-mentioned interaction activities is the value of 1. In this pattern the context-independency of Service BPEL Process is the value of 0.8. The details of calculation are as follows:

$$C_{BPEL}(BPEL) = \left[C_{<recieve>} + \left[(0.5 \times C_{<invoke>}) + (0.5 \times 0)\right] + C_{<invoke>}\right] \times \frac{1}{R_{client}}$$

$$C_{BPEL}(BPEL) = \left[1 + \left[(0.5 \times 1) + (0.5 \times 0)\right] + 1\right] \times \frac{1}{2} = 1.25$$

$$CI_{BPEL} = \frac{1}{1.25} = 0.8$$

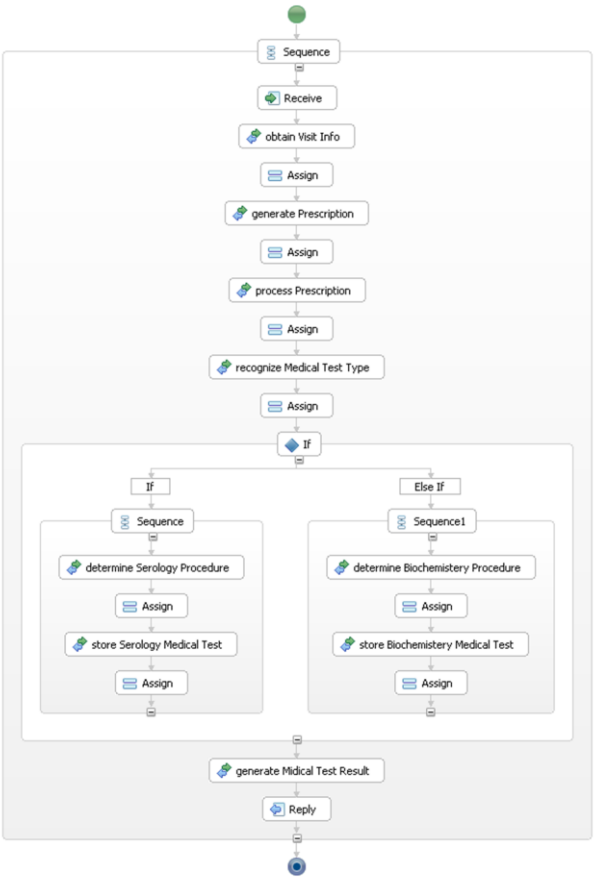

Figure 2: fulfillPatientMedicalTest BPEL process, Version 1.

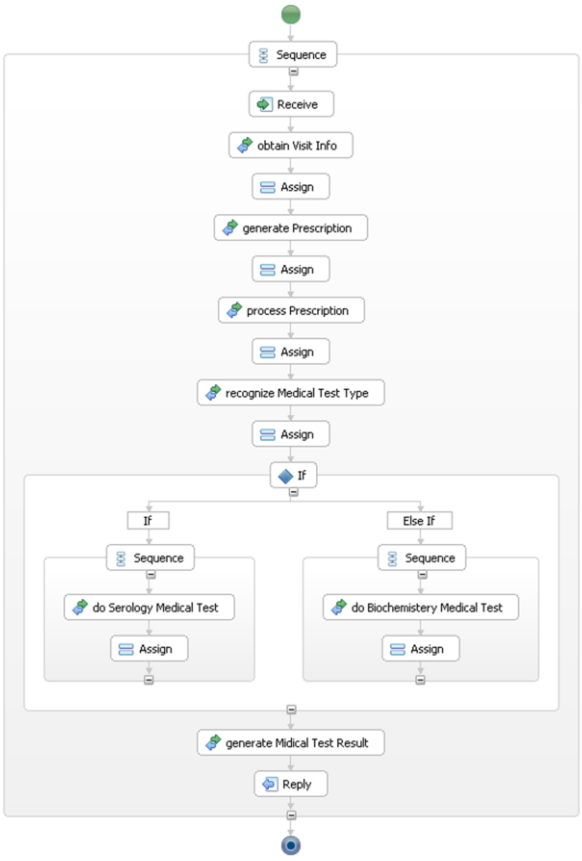

Figure 3: fulfillPatientMedicalTest BPEL process, Version 2.

In the same manner, different patterns could be evaluated even though they may get different values.

Evaluation

Exemplary Scenario

In order to elaborate more on how the proposed metric works, we apply it to one of the important healthcare processes, "fulfillPatientMedicalTest" which has been designed at different versions, Figure 2 and Figure 3. For readability reasons, the BPEL processes are modelled and represented through Eclipse BPEL Designer plug-in [26].

These BPEL processes are derived from the business processes of Iranian Hospitals and Laboratories. The service is triggered by receiving the information of patient's visit. After storing visit information the physician prescription is generated. In the next step the prescription is processed and required tests are recognized. Based on the type of test either biochemical or serological the procedure of test is determined and thereafter the required information is stored. Finally the test results are generated and delivered to the patient.

These two versions are actually the different designs of the same BPEL Process. Indeed, the difference arises from the diverse granularity of their partners. To calculate context-independency of the given BPEL processes, firstly we require the coupling value of interaction activities. Table 1 denotes the required values. To illustrate more, PatientRequest comprises PatientID and Request with String data type. Similarly, VisitInfo is a complex type with sequence behaviour in it.

VisitInfo consists of three primitive types including ID, VisitDate, and Medications with String data types.

Table 1: the coupling values of interaction activities in sample BPEL process Version 1 and 2.

BPEL process	Interaction activity	Operation	Message	C_{ia}
BPEL process version 1	<receive>	initiate	PatientRequest	2
	<Invoke>	obtainVisitInfo	VisitInfo	3
	<Invoke>	generatePrescription	Prescription	7
	<Invoke>	processPrescription	Prescription	7
	<Invoke>	recognizeTestType	TestType	5
	<Invoke>	determinSerologyPro	SerologyProcedure	8
	<Invoke>	storeSerologyTest	SerologyTest	12
	<Invoke>	determinBiochemisteryPro	BiochemisteryProcedure	6
	<Invoke>	storeBiochemisteryTest	BiochemisteryTest	9
	<Invoke>	generateTestResult	TestResult	15
	<reply>	response	TestResult	15
BPEL process version 2	<receive>	initiate	PatientRequest	2
	<Invoke>	obtainVisitInfo	VisitInfo	3
	<Invoke>	generatePrescription	Prescription	7
	<Invoke>	processPrescription	Prescription	7
	<Invoke>	recognizeTestType	TestType	5
	<Invoke>	doSerologyTest	SerologyTest	12
	<Invoke>	doBiochemisteryTest	BiochemisteryTest	9
	<Invoke>	generateTestResult	TestResult	15
	<reply>	response	TestResult	15

Now, we could calculate context-independency of given BPEL processes as follows:

- Context-independency of first version:

$$C_{BPEL}(BPEL) = \left[2+3+7+7+5+\left[(\frac{1}{2}\times(8+12)) + (\frac{1}{2}\times(6+9)) \right] +15+15 \right] \times 6 = 429$$

$$CI_{BPEL} = \frac{1}{429} \approx 0.002$$

- Context-independency of second version:

$$C_{BPEL}(BPEL) = \left[2+3+7+7+5+\left[(\frac{1}{2}\times 12) + (\frac{1}{2}\times 9) \right] +15+15 \right] \times 6 = 387$$

$$CI_{BPEL} = \frac{1}{387} \approx 0.003$$

As the computations denote, context-independency value for first version of fulfillPatientMedicalTest process is less than the second one. However, one could question that why the BPEL process version 2 is in more appropriate level of context-independency. In fact, the second version is more modular and its partners are more functionally cohesive[23] in comparison with the first version. The concept of modularity refers to the point that, the partners in second version of given BPEL process represents a better separation of concerns, through enforcing logical boundaries between services. To be more specific, determineSerologyProcedure and storeSerologyMedicalTest in version 1 encapsulated in doSerologyMedicalTets in version 2. Such cohesiveness of a module contributes lots of software qualities such as flexibility, maintainability, reusability, reliability, etc [27]. Therefore, after context-independency analysis architects can reengineer the BPEL process to reduce the complexity and increase the flexibility, if needed.

Implementation

In order to automate metric calculation, the authors propose a recursive algorithm for BPEL process context-independency calculation, Figure 4. The input of our algorithm is a BPEL process file and its output is context-independency value.

The algorithm accepts indexes of the first and the last lines of the XML tags, and the current XML tag of the process as input. At first, the algorithm finds the most internal tag of the BPEL process and calculates its coupling. Calculation of the coupling value for each tag depends on the current and previous tags of the process. Then the algorithm moves backward to calculate coupling value of the whole process.

```
Func coupling (start_curser, end_curser, current_tag)
BEGIN
        No=0;
        Branch=0;
        Statement=0;
        for (i=start_curser; i<=end_curser;  ++i)
                tag = read (ith line);
                case tag in
                        '<sequence>')
                                if (current_tag='switch' or current_tag='pick')
                                        Branch++;
                                j=find_close_tag();
                                statement=statement + coupling( i, j, 'seq');
                                i = j;
                        '<switch>')
                                if (current_tag='switch' or current_tag='pick')
                                        Branch++;
                                j=find_close_tag();
                                statement=statement + coupling( i, j, 'switch');
                                i = j;
                        '<pick>')
                                if (current_tag='switch' or current_tag='pick')
                                        Branch++;
                                j=find_close_tag();
                                statement=statement + coupling( i, j, 'pick');
                                i = j;
                        '<flow>')
                                if (current_tag='switch' or current_tag='pick')
                                        Branch++;
                                j=find_close_tag();
                                statement=statement + coupling( i, j, 'flow');
                                i = j;
                        '<while>')
                                if (current_tag='switch' or current_tag='pick')
                                        Branch++;
                                j=find_close_tag();
                                statement=statement + n * coupling( i, j, 'while');
                                i = j;
                        '<invoke>')
                                if (current_tag='switch' or current_tag='pick')
                                        Branch++;
                                        Statement++;
                                No= IVDF + OVDF;
                                i = j;
                        '<on message>' or '<receive>' or '<reply>' )
                                if (current_tag='switch' or current_tag='pick')
                                        Branch++;
                                        Statement++;
                                No= VDF;
                                i = j;
                        Break;
                end case;
        end for
        if (current_tag='switch')
                statement= statement/Branch;
        elif (current_tag='pick')
                if (state = '<on alarm>'
                        statement= statement/Branch;
                else
                        statement= statement/Branch + VDF;
        elif (current_tag='flow')
                if (synchronize)
                        NL= find_number_of_links();
                        statement= statement + NL;
                else
                        statement= 1/2 (statement ² + statement);
        else
                statement= statement + No;
        return statement;
END
```

Figure 4: A recursive algorithm pseudo code for BPEL Process coupling measurement.

```
Func Context_Independency (start_curser, end_curser)
BEGIN
      CI=0;
      R=0;
      for (i=start_curser; i<=end_curser; i++)
            r= role of ith tag;
            R= R + 1/r;
      Coupling= Coupling(start_curser, end_curser, <process>) * R;
      CI= 1 / Coupling;
      return CI;
END
```

Figure 5: A pseudo code for BPEL Process context-independency measurement.

To calculate the context-independency of the process, coupling factor should be multiplied by partner link roles factor and finally be divided by 1. The illustrated algorithm in the Figure 5 calculates context-independency of BPEL process.

Validation

Theoretical Validation

In this section, we aim to theoretically validate the proposed metrics using the property-based framework of Briand et al. [28]. This framework is a mathematical generic framework that introduces some intuitive properties for salient concepts of software engineering such as complexity, coupling, cohesion, and size, through which researchers and practitioners could analyze and validate the theoretical grounds of their measures irrespective of a specific development paradigm. Since the context-independency metric is founded on coupling metrics, we examine the proposed BPEL process coupling metric (C_{BPEL}) against five coupling properties including Nonnegavity, Null Value, Monotonicity, Merging of Modules, and Disjoint Module Additivity.

Property 1: Nonnegativity: This property is satisfied since for a given BPEL process the value of coupling metric is equal to zero when the process does not have any variable data field or some positive value representing the coupling between a BPEL process and its partners. Therefore, it will never be negative under any circumstances.

Property 2: Null Value: Since each BPEL process has at least one client partner link, hence there is not any process with null relationship. As a result, in reality this property is not applied to BPEL processes. However, this property is theoretically satisfied since C_{BPEL} for a given element will be null in the case where the element does not have any relationships.

Property 3: Monotonicity: This property is satisfied because coupling value of a BPEL process cannot be decreased by adding more relationships between this element and other existing or new partners.

Property 4: Merging of Modules: This property is also satisfied since if we merge the functionality of a given partner (for instance, a web service) in a BPEL process the coupling value of both elements will decrease or in the worse case hold the same value and will not increase.

Property 5: Disjoint Module Additivity: This property is also satisfied since in the proposed coupling metrics multiple connections or dependencies to the same partner are counted. In other words, the coupling of a BPEL process obtained by merging two unrelated ones is equal to the sum of the couplings of the two original processes, because in the worst case there are some common partners for two processes but we count all the connections to the partners. As a result, the coupling of disjoint elements will be additive.

As proposed coupling metrics satisfies all the prescribed properties for the corresponding attribute, hence it can be considered as a valid characterization of coupling and also context-independency of a BPEL process.

Empirical Validation

Although the proposed metrics theoretically are valid, this does not ascertain the predictive power of the metrics in terms of an empirical relationship between them and the context-independency that they supposed to predict.

This section presents a controlled experiment that authors have conducted to empirically validate the proposed metrics.

Empirical studies can be achieved by either controlled experiments or case-studies [29]. However, due to difficulties in conducting industrial investigation in the emerging paradigms such as SOC, we followed the first approach of empirical evaluation. Moreover, it has been suggested that the internal validity of the study should be determined prior to establishing external validity and controlled experiments provide better support for controlling instrumentation effects that can affect the internal validity of the study [29]. Additionally, Zelkowitz and Wallace [30] and Basili et al. [31] stress the importance of using experimental models for validating metrics. For the experiment to be successful it needs to be wisely constructed and executed. Therefore, we have followed some suggestions, provided by Perry et al. [32] and Mendonca et al.[33], about the structure and the components of a suitable empirical study. To perform an experiment, several steps have to be taken in a certain order. An experiment can be divided into the following main activities [33]: goals of the study, hypotheses, experimental design, threats to validity, data analysis and presentation, results and conclusions. In the remainder of this section we will explain how we have performed each of the above-mentioned activities.

Goals of the study

The main goals of this study are defined according to the GQM framework [33] as follows:

"Analysing the proposed context-independency metrics to evaluate their predictive capability with respect to the design-level estimation of the independency of the BPEL process to their contexts."

Hypothesis Formulation

An important aspect of experiment is to know and maintain in a formal way what we intend to evaluate. Hypotheses are essential as they state the research questions we are asking. Authors present two hypotheses as follows:

Abstract Hypothesis: "The context-independency (CI) metric is a good and accurate metric to evaluate the context-independency of a BPEL processes."

Concrete Hypothesis: "There is a significant correlation between the CI metric and the expert's rating of the context-independency of BPEL processes."

Experimental Protocol

After the research context and the hypotheses formulation, the design of the study took place. The study design is a detailed plan for collecting the data that will be used to test the hypotheses. This phase also explains how the experiment was conducted and has several components that are fully described in order to provide useful information for future replications [34]:

Variable selection

Typically, in empirical studies there are two kinds of variables (dependent and independent) that their cause and effect should be evaluated by testing the hypothesis with appropriate techniques. In this research they are as follows:

a) The independent variable is the structure of BPEL processes.

b) The dependent variable is the context-independency of processes which varies when the structure of BPEL processes changes.

Expert selection

Our experts were students of the Electrical and Computer Engineering Faculty enrolled in the Master and PhD program in Computer Engineering at the Shahid Beheshti University, Tehran, Iran. Since the desired population for the study was rare and very difficult to locate, they were selected based on purposive sampling [35].

Twenty experts were selected based on the evaluations of their lecturers. Half of the participants were male in the 24 –50 age group and the rest were female in the 23-29 age groups. Moreover, all participants were volunteers who had interest in software engineering research.

Most of them had industrial experience in several areas, but none had experience with business process management systems. By the time the experiment was done, all the students had taken a 50 hours course on ULS (Ultra Large-Scale Systems) with emphasis on service-oriented systems, business processes modeling, and service composition, therefore, gained experience in the design and development of services specially business services. To enhance their knowledge about service modeling a group-based training session was carried out before doing experiment. This

session consisted of an introduction to BPEL language, its constructs, and the quality attributes of a BPEL process.

Experiment design

The objects to be rated were business services (i.e. BPEL process) graphically designed with the Eclipse BPEL designer. The independent variable was measured using the CI metric presented in section 4. The dependent variable was measured according to expert's ratings.

Authors prepared the required material in order to gather participant's judgment based on the rates. The material consisted of 70 professionally designed BPEL processes[1] of the different universe of discourses such as university and core banking with different structural characteristics and degree of complexity.

The participants were told how to carry out the experiment. In order to make the experience and knowledge of the participants more comparable, we made 10 groups out of the 20 participants. They could use unlimited time to make a consensus based on their judgments. Our key aim to give them unlimited time was based on the fact that we do not intend to rush their consensus. During the workshop they felt free to discuss enough to make a consensus. However, since the group's experiences were comparable they finished rating after 3 hours and deliver their tasks around the same time. Table 2 denotes the experts' profile. In order to gather more precise rating, authors decided to design a professional questionnaire and utilized it during data collection. We collected all the data include experts' rating and the measurements which were calculated by means of the CI metric.

Table 2. Participant groups and associated profiles.

Group Index	No of Members	Profiles
1	2	Master Students in software engineering
2	2	Master Students in software engineering
3	3	Master Students in IT
4	2	Master Students in IT
5	2	Master and PhD Students in software engineering
6	2	Software Engineering Practitioners
7	2	Master Students in software engineering
8	3	Master Students in software engineering
9	1	PhD student in Software Engineering
10	1	Professor in software engineering

Threats to Validity

Threats to validity may limit our ability to interpret or draw conclusions from the results. In this section various threats to validity (construct, internal, and external validity) and the way we attempted to alleviate them will be discussed.

Construct validity

[1] These services are available at ASER Group Website (http://aser.sbu.ac.ir/soca/).

The independent variable that estimates the context-independency of processes can be considered constructively valid because they are defined in a formal manner and also theoretically validated in Section 7.1.

All the measurements of the dependent variable were subjective and based on the perception of the experts. As the experts involved in this experiment had medium experience in service design and they attain minimum knowledge of BPEL process design, hence their ratings can be considered significant.

Internal validity

We have considered different aspects that could threaten the internal validity of the study such as differences among experts, precision of experts' ratings, learning effects, fatigue effects, and experts' incentive.

- Differences among experts: We grouped the participant as described before; therefore error variance due to differences among participants was reduced. Additionally, purposive sampling was employed to select participants with comparable knowledge and experience. Experts involved in this experiment had medium experience in service design and they attain minimum knowledge of BPEL process design, their ratings can be considered significant.
- Practice [learning and fatigue] effects: Since the experiment materials were in the different universe of discourses with different structural characteristics and degrees of complexity and the participants were in the small groups discussing on the issues to reach an appropriate consensus, it is very unlikely that any potential learning and fatigue affects the data.
- Instrumentation effects: One threat to internal validity is processes that have "while" construct. Since in design time designer could not accurately estimate the number of potential loop iterations, it is a potential threat to the validity of dependent variable.
- Anticipation effects: The participants were not told about the hypothesis that we wanted to test in order to ensure that expectations about specific levels of treatment did not influence their rates.

External validity

One threat to external validity is participant selection. This threat can limit the ability to generalize the results to settings outside the study. The experts were Master and PhD students that had recently taken a 50 hours course on ULS gaining a sufficient experience in the design and development of services. In order to extract a final conclusion that can be generalized, it is necessary to replicate this experiment with a more diversified number of experts, including practitioners and designers with more experience.

- Experimental materials and Environment: The materials of our study are represented with BPEL and WSDL which both of them were standard and are utilized in industrial environment. Additionally, they were representative of real world composite services in terms of size, and the complexity of the processes.

We have described the experiment. The following step is to present the descriptive and statistical analysis that was carried out to validate the CI metric. This was done by taking the data obtained concerning the dependent variables, to determine the feasibility of using the CI metric to estimate the context-independency of BPEL processes.

Analysis of the Results

Since our experts rated BPEL processes using a numerical scale from 1 to 10, we have selected quantitative analysis to draw conclusions from our data. Moreover, the qualitative analysis was done in conjunction with a statistical analysis.

Descriptive Statistics

The descriptive statistics are presented in Tables 3 for each independent and dependent variable.

Table 3: Descriptive Statistics.

		Mean	Std. Deviation	Variance	N
Gr	1	3.8714	2.29632	5.273	70
	2	2.9714	1.86479	3.477	70

	3	3.2000	1.49976	2.249	70
	4	2.9143	2.15852	4.659	70
	5	1.9714	1.46427	2.144	70
	6	3.2429	2.34944	5.520	70
	7	4.1714	2.27763	5.188	70
	8	4.3714	2.55474	6.527	70
	9	4.0000	2.21981	4.928	70
	10	3.1429	1.90564	3.631	70
CI		.3491	.18013	.032	70

Hypothesis Testing: Statistical Analysis

In this section, authors want to determine if any correlation exists between experts' ratings and the proposed CI metric. The first step in the correlation analysis was to ascertain whether the distribution of the data was normal, so the Kolmogorov-Smirnov test was applied. As we obtained that the distribution was not normal, we decided to use a non-parametrical statistical test, namely the Spearman Rank-Difference Correlation Coefficient [36] with two levels of significance of $\alpha1=0.01$ and $\alpha2=0.05$, which indicates the probability of rejecting the null hypothesis when it is certain (type I error). That is a confidence level of 99% and 95% exists respectively. The Spearman r_S is a non-parametric statistics used to show the relationship between two variables which are expressed as ranks (the ordinal level of measurement). The correlation coefficient is a measure of the ability of one variable to predict the value of another variable. Using Spearman's correlation coefficient, the CI metric was correlated separately to the different experts' rates of context-independency. In our experiment the null hypothesis was:

H_0: "there is no significant correlation between the CI metric and the experts' rating of context-independency".
H_1: "there is a significant correlation between the CI metric and the experts' rating of context-independency".

The probability that the null hypothesis would be erroneously rejected was controlled with two confidence levels: $\alpha1=0.01$ and $\alpha2=0.05$.

Table 4.The Spearman rank-difference correlation coefficient between the group of participants' ratings and the values given by the CI metric.

		r_s	$\alpha1$	$\alpha2$
Group Index	1	-.641	Reject H0	Reject H0
	2	-.575	Reject H0	Reject H0
	3	-.245	Accept H0	Reject H0
	4	-.739	Reject H0	Reject H0
	5	-.587	Reject H0	Reject H0
	6	-.677	Reject H0	Reject H0
	7	-.609	Reject H0	Reject H0
	8	-.678	Reject H0	Reject H0
	9	-.648	Reject H0	Reject H0
	10	-.606	Reject H0	Reject H0
Average		-.738	Reject H0	Reject H0

The analysis performed on the collected data led to some interesting results. Table 4 denotes the summary statistics describing the Spearman rank-difference correlation coefficient between experts' ratings and the values given by the CI metric. For each experiment, the correlation coefficient, r_S, the significance of each coefficient and the number of cases with non-missing values are also displayed in the table. The absolute value of the correlation coefficient indicates the strength, with larger absolute values indicating stronger relationships. The significance level (or p-value) is the probability of obtaining results as extreme as the one observed. If the significance level is very small (less than 0.05) then the correlation is significant and the two variables are linearly related.

Based on the data from Table 4 and taking $\alpha 1$ into consideration, correlation is significant at the 0.01 level for 90% of the groups; therefore, the null hypothesis was rejected with 99% confidence. Taking $\alpha 2$ into consideration, all the values of r_S shows significant correlation; therefore, the null hypothesis is also rejected with the second confidence level that is equal to 95%.

Power Analysis of the Hypothesis Test

A test without sufficient statistical power will not produce enough information to convince the researcher regarding the acceptance or rejection of the null hypothesis [37]. Since we are using small samples, authors provide details of power analysis in this section to interpret the results. The power of the nonparametric tests is determined by appropriate parametric tests [37]. Therefore, we have used Pearson's *r* [38] for the Spearman Rank Correlation that we have utilized for testing the hypothesis. The G*Power 3 [39] tool was used to conduct a power analysis in order to compute the achieved power of the statistical test used in the study. Given: i) observation size of 70 from 10 group of participants that produce 700 data points; ii) level of α= 0.01; and iii) the effect size 0.6; the achieved power is 0.81 that is considerably fine [37] in software engineering studies. However, by the effect sizes lower than 0.6 the power would dramatically shrinks. Thereby suggesting that the statistical tests could produce a Type II error [37] (i.e. fail to reject a null hypothesis when it is in fact false) when the test do not exhibit large effect sizes.

Descriptive Analysis

After analyzing the gathered data, we concluded that the obtained results reveal that there exists a high correlation between the CI metric and the expert's rating of context-independency as the scatter diagram in Figure 6 demonstrates. This leads us back to our original goal which was to demonstrate that the CI metric serves the purpose it was defined for, estimate the context-independency of BPEL processes. The obtained results are believable and there are no ambiguities in our interpretation. We also believe that no external elements have influenced our results. The diffusion of the experimental results and the way they are presented are relevant so that they can be put into use. Therefore, authors published the findings in this paper and we are also planning to develop a web-based system to allow other researcher to expand our experiment.

Our results recommend the use of the CI metric to create less coupled business services to environment, thus reducing the time spent reading and understanding services in order to adapt them to changing requirements. The context-independency estimation enables process managers and administrators to calculate the context-independency of BPEL processes generated by others. Architect can analyze the context-independency of a particular business service in development.

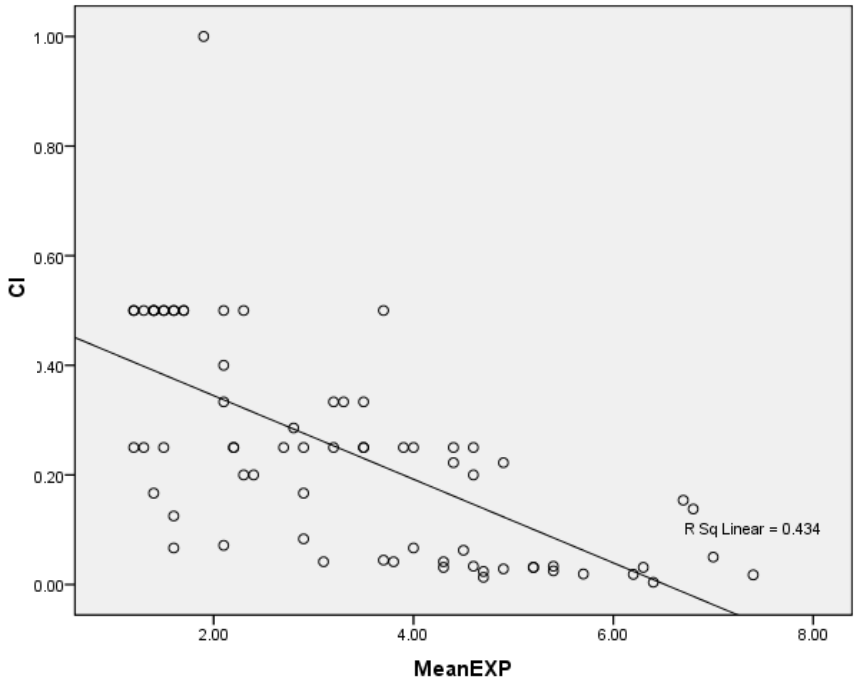

Figure 6: Scatter Plot of Average of rates and CI metric

Discussion

A BPEL process can be terminated normally or abnormally. Normal termination occurs when all activities complete. Abnormal termination occurs either when a fault occurs, or a process instance is terminated explicitly using the <terminate> construct [5]. In BPEL there are some activities through which process could handles abnormal behaviors.

Fault handlers, compensation handlers, and event handlers provide some facilities to manage the behavior of abnormal execution scenarios. To signal the fault to the clients, for example, <reply> construct for synchronous and <invoke> activity for asynchronous BPEL process are used. Based on the proposed formulas the coupling value of fault handlers can be calculated too. Nevertheless, in BPEL, faults can be sourced from WSDL faults, error conditions in run-time environment, and so on. Therefore, context-independency calculation in such cases is dependent on what happens in run-time environment, while the proposed metrics are supposed to be leveraged at analysis and design phase of SOA development before implementation takes place.

In the same manner, we could independently calculate the coupling value of fault handler since the syntax of the event handler section is similar to the syntax of the <pick> activity. However, suppose that an event handler allows the BPEL process client to cancel that at any time. Thus, we are not able to aggregate and contemplate them in calculation at design time.

Conclusions

In this article, authors proposed a metric for BPEL process context-independency analysis. Our approach for predicting BPEL process context-independency is based on the coupling value measurement of a BPEL process to its partners. The coupling value of a certain BPEL process is examined and quantified on the basis of measuring its interaction activities within structured activities. This paper also studied a certain BPEL process in detail which exhibit how the proposed metric works.

To validate our metrics, we collected a data set consisting 70 BPEL processes and also gathered the expert's rating of context-independency through holding a workshop. The obtained results reveal that there exists a high statistical correlation between the proposed metrics and the expert's judgment of context-independency.

Acknowledgement

The project has been partially funded by Shahid Beheshti University. This work was also supported, in part, by Science Foundation Ireland grant 03/CE2/I303_1 to Lero - the Irish Software Engineering Research Centre (www.lero.ie)

We would like to thank the involved computer science graduate students of Shahid Beheshti University and in particular Miss S. Khoshnevis for their supports.

We also would like to thank to anonymous reviewers of IEEE International Conference on Service-Oriented Computing and Applications (SOCA'09) and Service-Oriented Computing and Applications Journal, for their valuable comments.

References

[1] Kammer P J, Bolcer G A, Taylor R N, Hitomi A S, and Bergman M. (2000) "Techniques for supporting dynamic and adaptive workflow". Comput. Supported Coop. Work, 9(3-4):269–292, 2000.

[2] Reichert M and Rinderle S (2006) "On Design Principles for Realizing Adaptive Service Flows with BPEL"; Proc. EMISA. GI Lecture Notes in Informatics, LNI P-95, pp. 133-146.http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.2255&rep=rep1&type=pdf

[3] Cardoso J, (2007) "Complexity Analysis of BPEL Web Processes", Software Process Improvement and Practice, Wiley InterScience; 12, 35–49.

[4] Pressman R S, Software Engineering, a practitioner's approach, 5th Edition, McGrawHill, 2001.

[5] Juric M B, Mathew B, Sarang P, (2006) “Business Process Execution Language for Web Services”, Packt Publishing, Birmingham, B27 6PA, UK, Second edition.

[6] Perepletchikov M, Ryan C, Frampton K, (2006);”Towards the Definition and Validation of Coupling Metrics for Predicting Maintainability in Service-Oriented Designs”, OTM Workshops 2006, LNCS 4277, pp. 34 – 35, Springer-Verlag Berlin Heidelberg.

[7] Vanderfeesten I, Reijers H A, and van der Aalst W M (2008). Evaluating workflow process designs using cohesion and coupling metrics. *Comput. Ind.* 59, 5 (May. 2008), 420-437. DOI= http://dx.doi.org/10.1016/j.compind.2007.12.007

[8] Perepletchikov M., Ryan C, Frampton K, Tari Z (2007): “Coupling Metrics for Predicting Maintainability in Service-Oriented Designs”. In: Software Engineering Conference, ASWEC 2007, Melbourne, Australia, pp. 329–340.

[9] Stevens W, Myers G, and Constantine L, (1974) "Structured Design", IBM Systems Journal, vol. 13 (2).

[10] Sindhgatta R, Sengupta B, Ponnalagu K, (2009) Measuring the Quality of Service Oriented Design, ICSOC-ServiceWave 2009, LNCS 5900, pp. 485–499. Springer-Verlag Berlin Heidelberg.

[11] Pautasso C, Wilde E, (2009) “Why is the web loosely coupled? a multi-faceted metric for service design”. In: Proc. of the 18th World Wide Web Conference, Madrid, Spain.

[12] Xu T, Qian K, He X, (2006) “Service oriented dynamic decoupling metrics”. In Proc. the 2006 International Conference on Semantic Web and Web Services (SWWS'06), Las Vegas, USA, June 26-29, 2006, pp.170-176.

[13] Reijers H A and Vanderfeesten I T P, (2004) “Cohesion and Coupling Metrics for Work-flow Process Design”. In J. Desel, B. Pernici, and M. Weske, editors, International Conference on Business Process Management (BPM 2004), volume 3080 of Lecture Notes in Computer Science, pages 290-305. Springer-Verlag, Berlin.

[14] Reijers H A, (2003) “A Cohesion Metric for the Definition of Activities in a Workflow Process”. Proceedings of the Eighth CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design 2003, pages 116-125.

[15] Vanderfeesten I, Cardoso J, Reijers H A, (2007), “A weighted coupling metric for business process models”, The 19th International Conference on Advanced Information Systems Engineering (CAiSE Forum),11-15.

[16] Reijers H A, (2003) “Design and Control of Workflow Processes: Business Process Management for the Service Industry”. Lecture Notes in Computer Science 2617. Springer-Verlag, Berlin.

[17] Steghuis C, (2006), “Service Granularity in SOA Projects: A Trade-off Analysis”, M.Sc. Thesis, Business Information Technology, University of Twente, http://essay.utwente.nl/57339/1/scriptie_Steghuis.pdf. Accessed 14 June 2010.

[18] Papazoglou M, (2003) "Service-Oriented Computing: Concepts, Characteristics and Directions," in: Fourth International Conference on Web Information Systems Engineering, IEEE, Roma, Italy.

[19] Orton J D, and Weick K E, (1990) “Loosely Coupled Systems: A Reconceptualization”, Academy of Management Review 15 (2):203-223.

[20] Hurwitz J, Bloor R, Baroudi C, and Kaufman M, (2007) Service-Oriented Architecture for dummies, Wiley.

[21] Erl T, (2005) “Service-oriented architecture: concepts, technology, and design”, Prentice Hall.

[22] Fenton N E, and Pfleeger S L, (1998) “Software Metrics: A Rigorous and Practical Approach”, 2nd ed: PWS Course Technology Ptr.

[23] Papazoglou M P and Heuvel W J van den. (2006). Service oriented design and development methodology. *Int. J. Web Eng. Technol.* 2, 4 (July 2006), 412-442. DOI=10.1504/IJWET.2006.010423 http://dx.doi.org/10.1504/IJWET.2006.010423

[24] Oracle BPEL Process Manager Developer's Guide 10g Release 2 (10.1.2); BPEL processes common interaction patterns; http://download.oracle.com/docs/cd/B14099_19/integrate.1012/b14448/interact.htm. Accessed 20 July 2010.

[25] Barros A, Dumas M, ter Hofsted, Service Interaction patterns; Joint initiative by SAP and Queensland University of Technology, co-funded by Queensland State Government. http://math.ut.ee/~dumas/ServiceInteractionPatterns/patterns.html. Accessed 20 July 2010.

[26] Eclipse BPEL Designer Plug-in: http://www.eclipse.org/bpel/. Accessed 26 July 2010.

[27] Bowen T P, Post J V, Tsai J, Presson P E, Schmidt R L, (1983) Software Quality Measurement for Distributed Systems, Guidebook for Software Quality Measurement, RADC-TR-83-175, Vol. II, Final Technical Report, Rome Air Development Center, Air Force Systems Command, Griffis Air Force Base, NY.

[28] Briand L C, Morasca S, Basili V R, (1996) "Property-Based Software Engineering Measurement", IEEE Transactions on Software Engineering, vol. 22 (1), pp. 68-86.

[29] Perepletchikov M, Ryan C, (2010) "A Controlled Experiment for Evaluating the Impact of Coupling on the Maintainability of Service-Oriented Software," IEEE Transactions on Software Engineering, 01 Jun. 2010. IEEE computer Society Digital Library <http://doi.ieeecomputersociety.org/10.1109/TSE.2010.61>

[30] Zelkowitz M V, and Wallace D R, (1998) "Experimental Models for Validating Technology". IEEE Computer. 31(5): p. 23-31

[31] Basili V R, Briand L C, Melo W L, (1996) "A Validation of Object-Oriented Design Metrics as Quality Indicators" IEEE Transactions on Software Engineering, 22(10)

[32] Perry D E, Porter A A, and Votta L G, (2000) Empirical Studies of Software Engineering: A Roadmap, in the Future of Software Engineering, A. Finkelstein, Editor. 2000, ACM Press. ISBN 1-58113- 253-0.

[33] Mendonca M G and Basili V R, (2000) "Validation of an Approach for Improving Existing Measurement Frameworks", IEEE Transactions On Software Engineering, VOL. 26, NO. 6, JUNE 2000.

[34] Juristo N, Vegas S, (2010) "The role of non-exact replications in software engineering experiments", Empir Software Eng, DOI 10.1007/s10664-010-9141-9.

[35] Shaughnessy J, Zechmeister E B, and Zechmeister J S, (2005) "Research Methods in Psychology", 7th ed.: McGraw-Hill Humanities Social.

[36] Siegel S, Castellan N J Jr, (1988) "Nonparametric Statistics for the Behavioural Sciences", McGrawHill.

[37] Dyba T, Kampenes V B, Sjoberg D I K, (2006) A systematic review of statistical power in software engineering experiments, Information and Software Technology , Volume 48, Issue 8, August 2006, Pages 745-755.

[38] Hays W L, (1994) Statistics, fifth ed., Harcourt Brace, New York.

[39] Faul F, Erdfelder E, Lang A, and Buchner A, (2007) "G*Power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences," Behavior Research Methods, vol.39 (2), pp. 175-191.